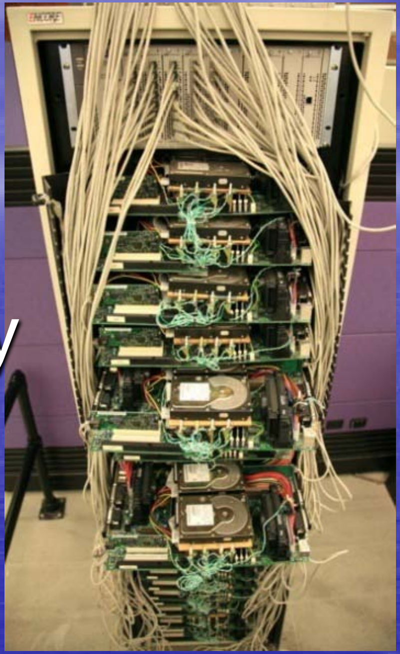
Reliable Distributed Storage
A Research Perspective

Idit Keidar
Technion

UnDistributed Enterprise Storage *(What I Won't Talk About Today)*

- Expensive
- Needs to be replaced to scale up
- Direct fiber access
  - But trouble if multiple machines access same data
  - Use server (bottleneck) or High-Availability Cluster
- Maybe bullet-proof
  - But single-point-of-failure
  - Get Disaster Recovery solution

Alternatives to Enterprise Storage *(I'll Talk About Today)*



Google's 1st server

1. Distributed Storage

   Made up of many cheap, low-reliability storage nodes

   – Achieving reliability, consistency

   – The reconfiguration challenge

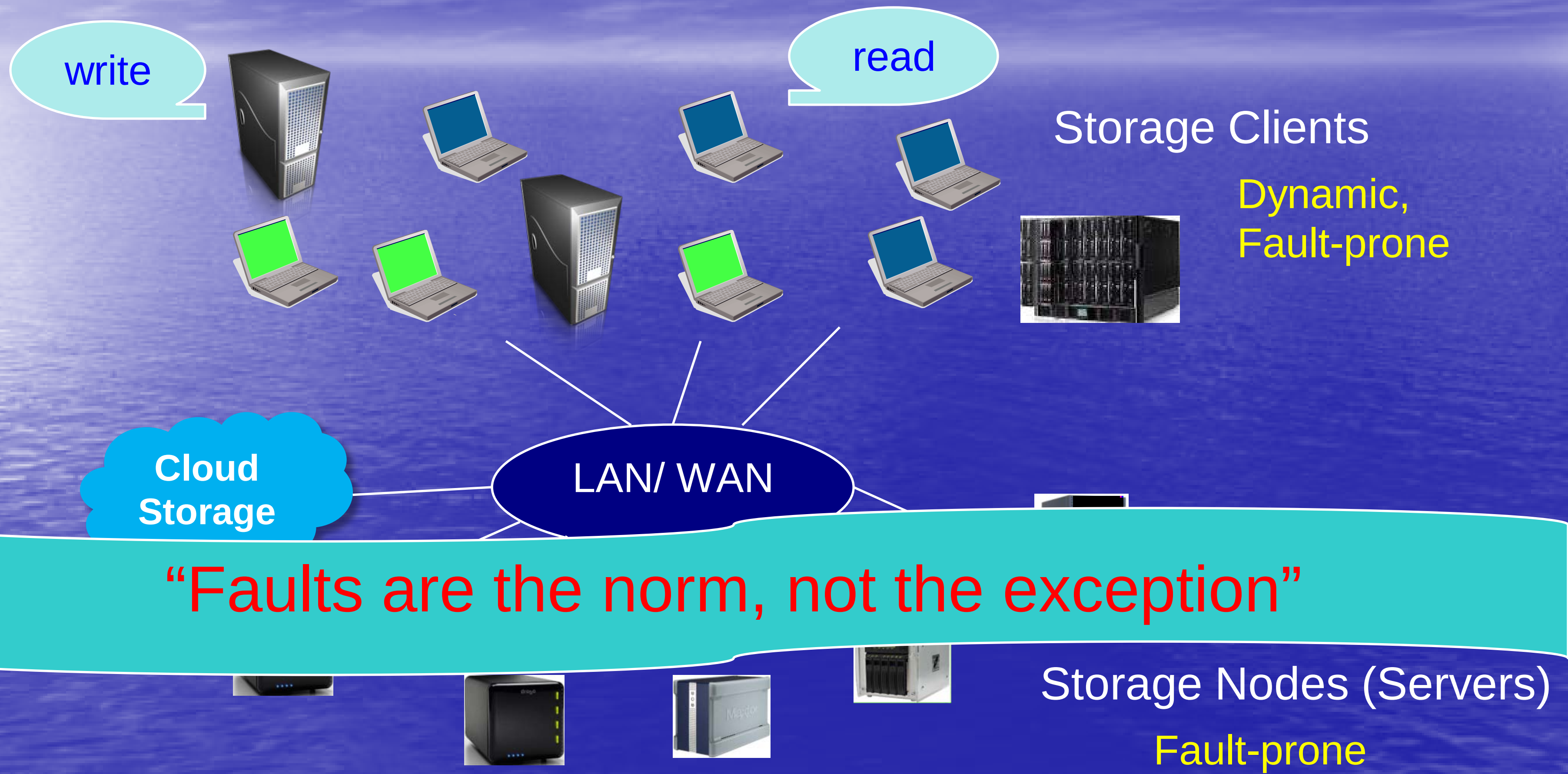
2. Cloud Storage

   – Can we trust the cloud to ensure reliability and consistency?

A Short Introduction to Reliable Distributed Storage

Chockler, Guerraoui, Keidar, Vukolic:
Reliable Distributed Storage, IEEE Computer 2009

Distributed Storage Architecture

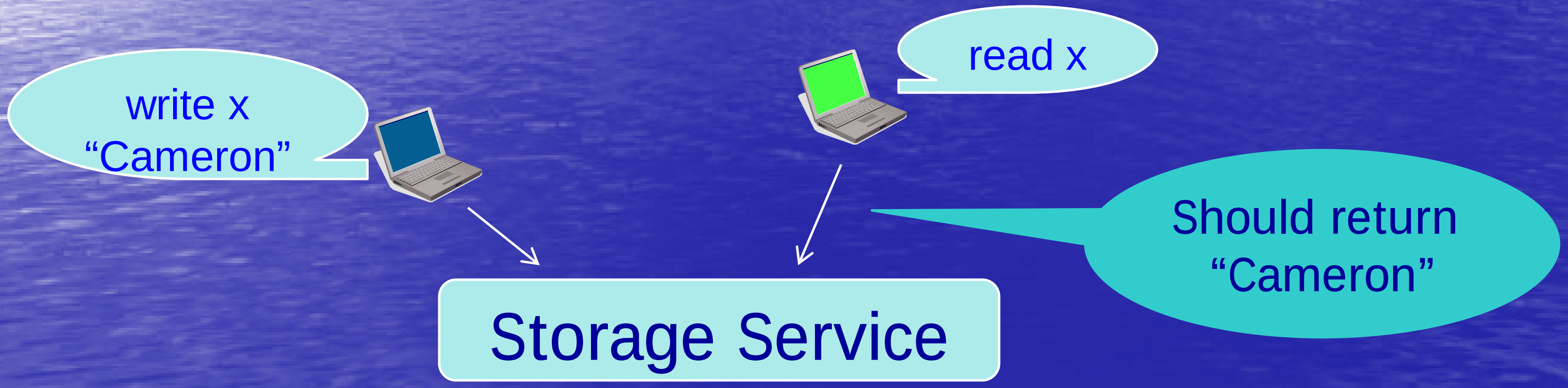


Getting Fault-Tolerance

- Replication
  - – Multiple copies (e.g., 3) of each data item
  - – Copies on distinct storage nodes
- Disaster recovery
  - – Copies geographically dispersed

Consistency

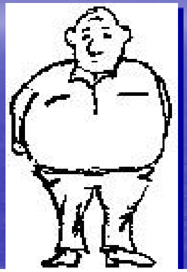
- Need to ensure that updates are reflected consistently in all copies
- Consistency means atomic operations

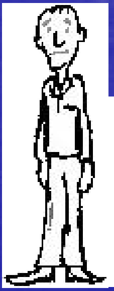


- Need Replica Coordination!

A Case for Data-Centric Replica Coordination

- Client-side code runs coordination logic
  - Communicates with multiple storage nodes
  - May be in middleware tier
- Simple storage nodes (servers)
  - Can be network-attached disks
    - ✓ Not necessarily PCs with disks
  - Simply respond to client requests
    - ✓ High throughput
  - Do not communicate with each other
    - ✓ No scalability limit

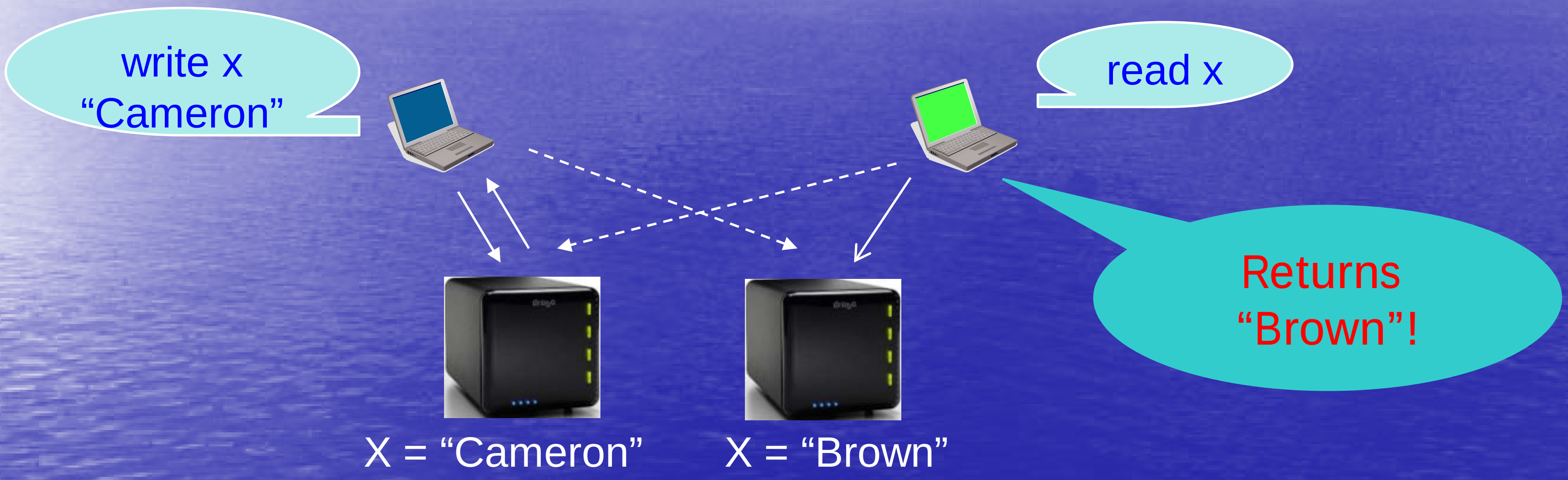




High Availability and Asynchrony

- Replication allows for high availability
- Client operations do not need to wait for all replicas
- Asynchronous communication

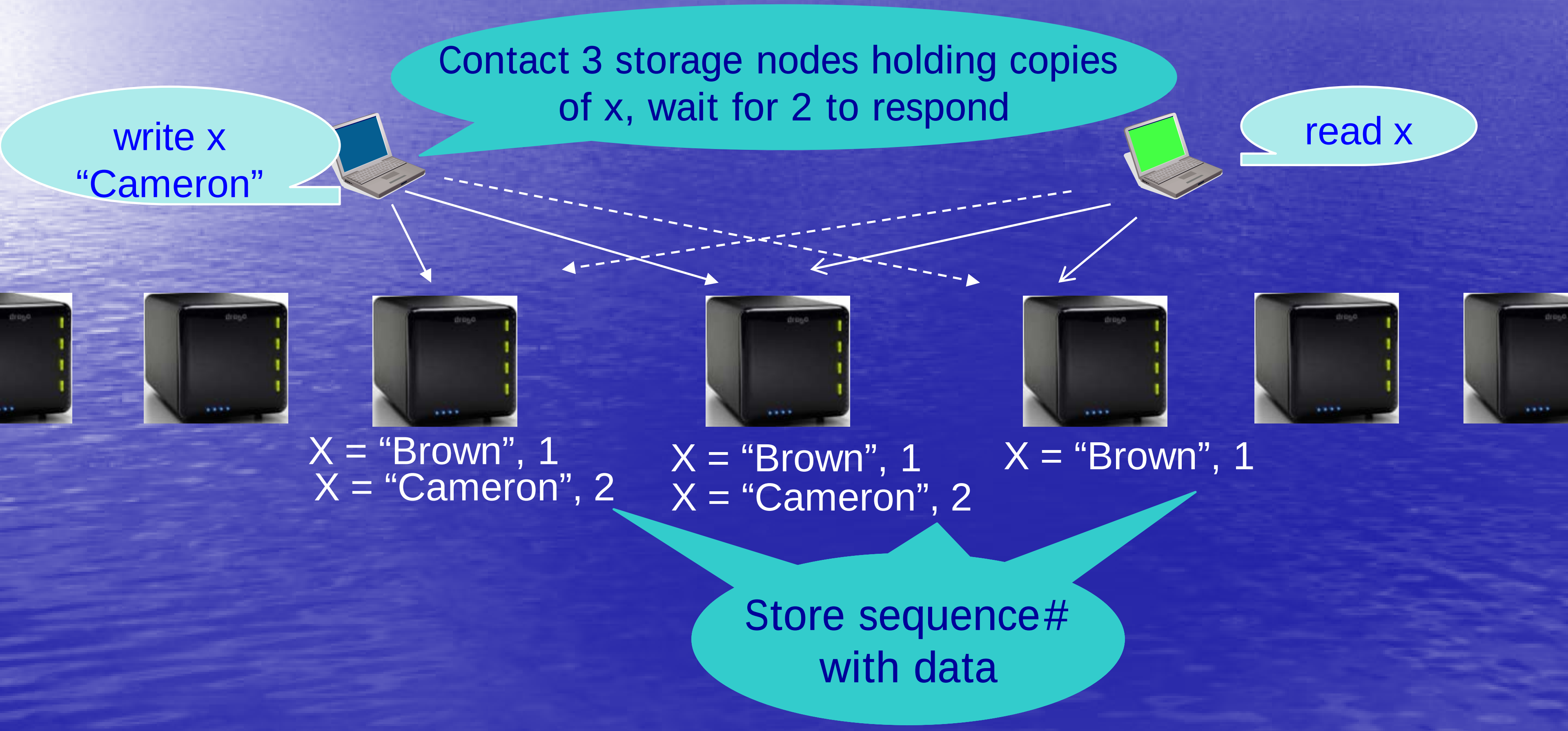
Two Copies Are Not Enough
With Asynchronous Communication



- Need to access a majority of copies
  - <u>Service availability</u>: when < half the copies fail

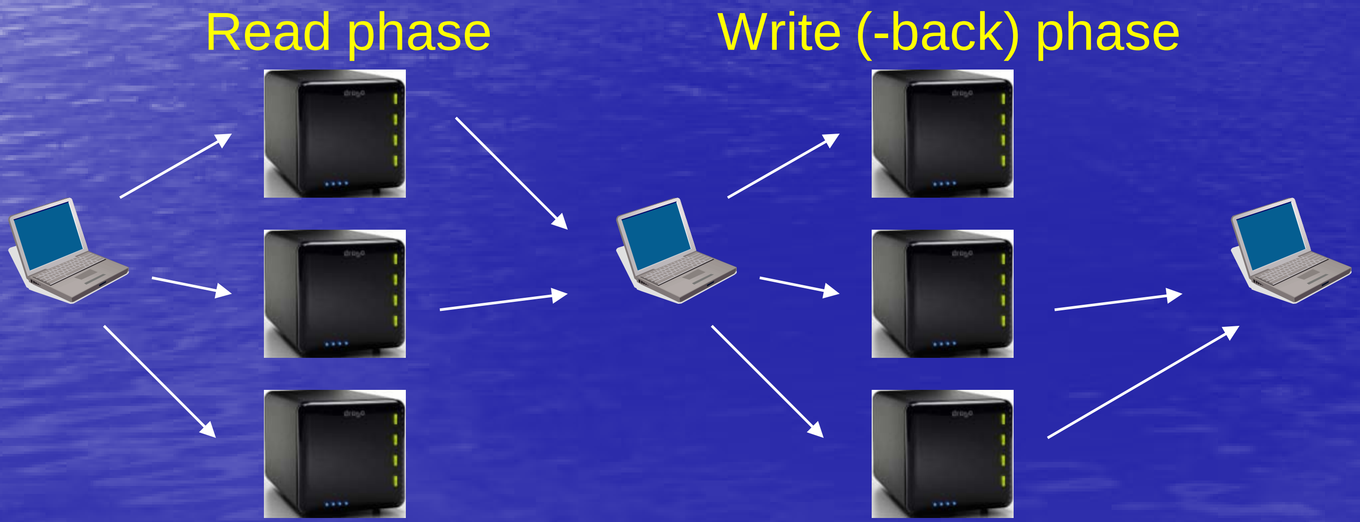
A Simple Reliable Distributed Storage Algorithm

- A-la ABD [*Attiya, Bar-Noy, Dolev* JACM 1995]



A Simple Reliable Distributed Storage Algorithm (Cont'd)

- Need to read before writing
  - To choose sequence#
- May need to write-back after reading
  - So next reader doesn't see older value (see paper)



Many Variants and Extensions in The Literature

- Implementations, system optimizations
  - FAB (HP Labs), Ursa Minor (CMU)
  - May use erasure-coding instead of full replicas to reduce storage blow-up
- Tolerating malicious faults, bugs, "Byzantine" faults

  Malkhi, Reiter: Byzantine Quorum Systems

  Abraham, Chockler, Keidar, Malkhi: Byzantine Disk Paxos

  …

Outline

1. Distributed Storage

   Made up of many cheap, low-reliability storage nodes

   – Achieving reliability, consistency

   – The reconfiguration challenge

2. Cloud Storage

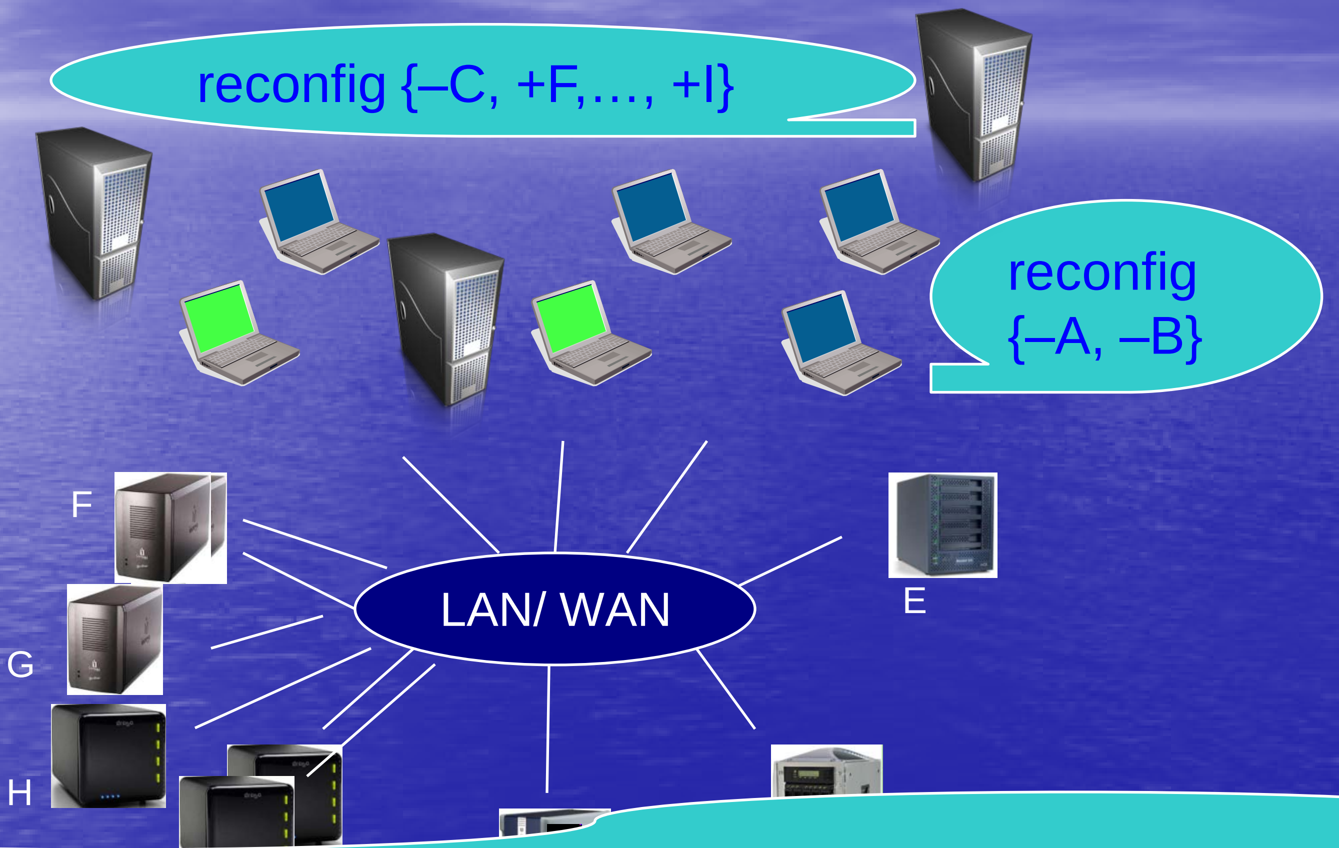
   – Can we trust the cloud to ensure integrity and consistency?

Dynamic (Reconfigurable) Distributed Storage

Aguilera, Keidar , Malkhi, Shraer:
Dynamic Atomic Storage Without Consensus, PODC'09

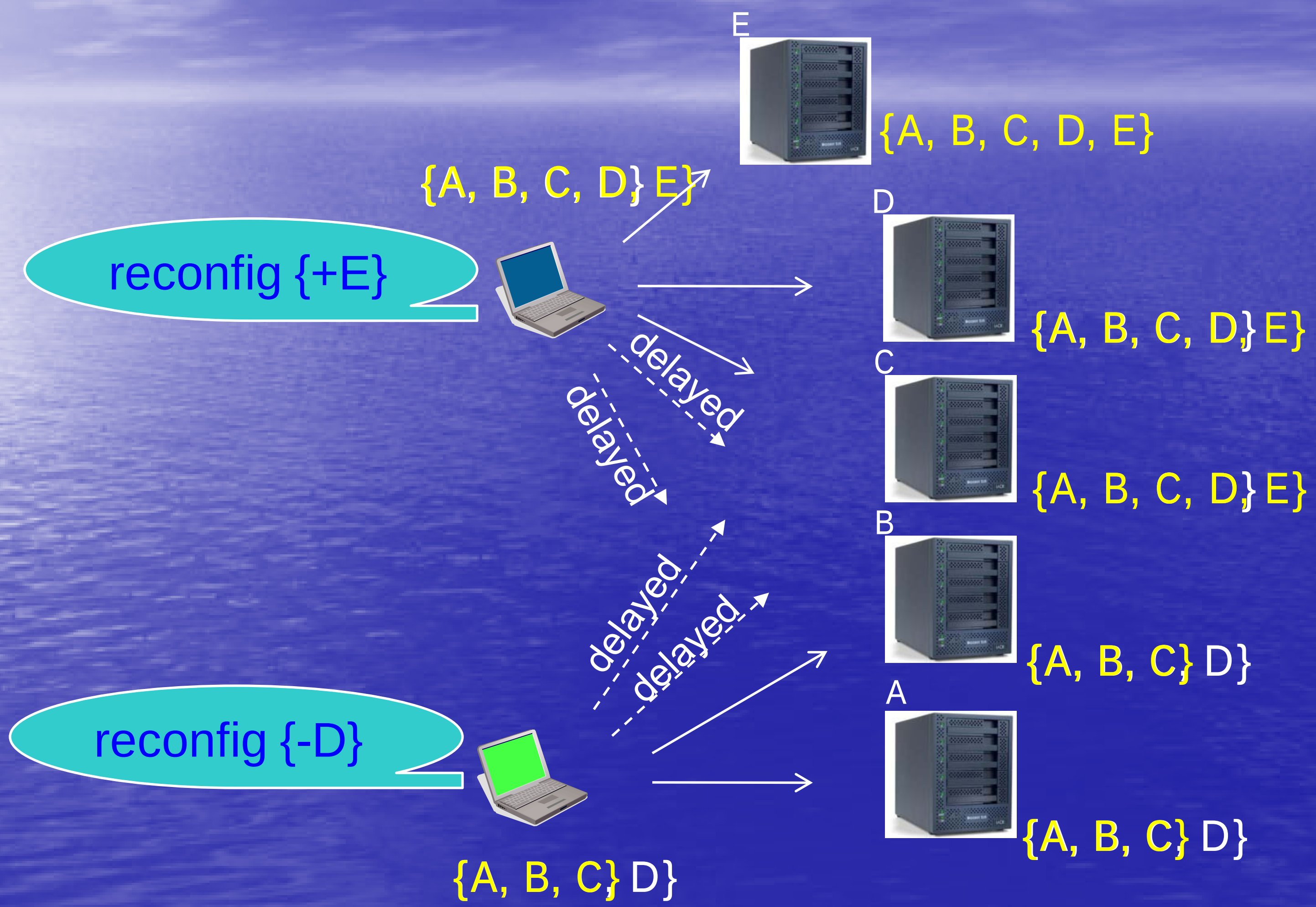
Shraer, Martin, Malkhi, Keidar:
Data-Centric Reconfiguration with
Network-Attached Disks

Real Systems Are Dynamic

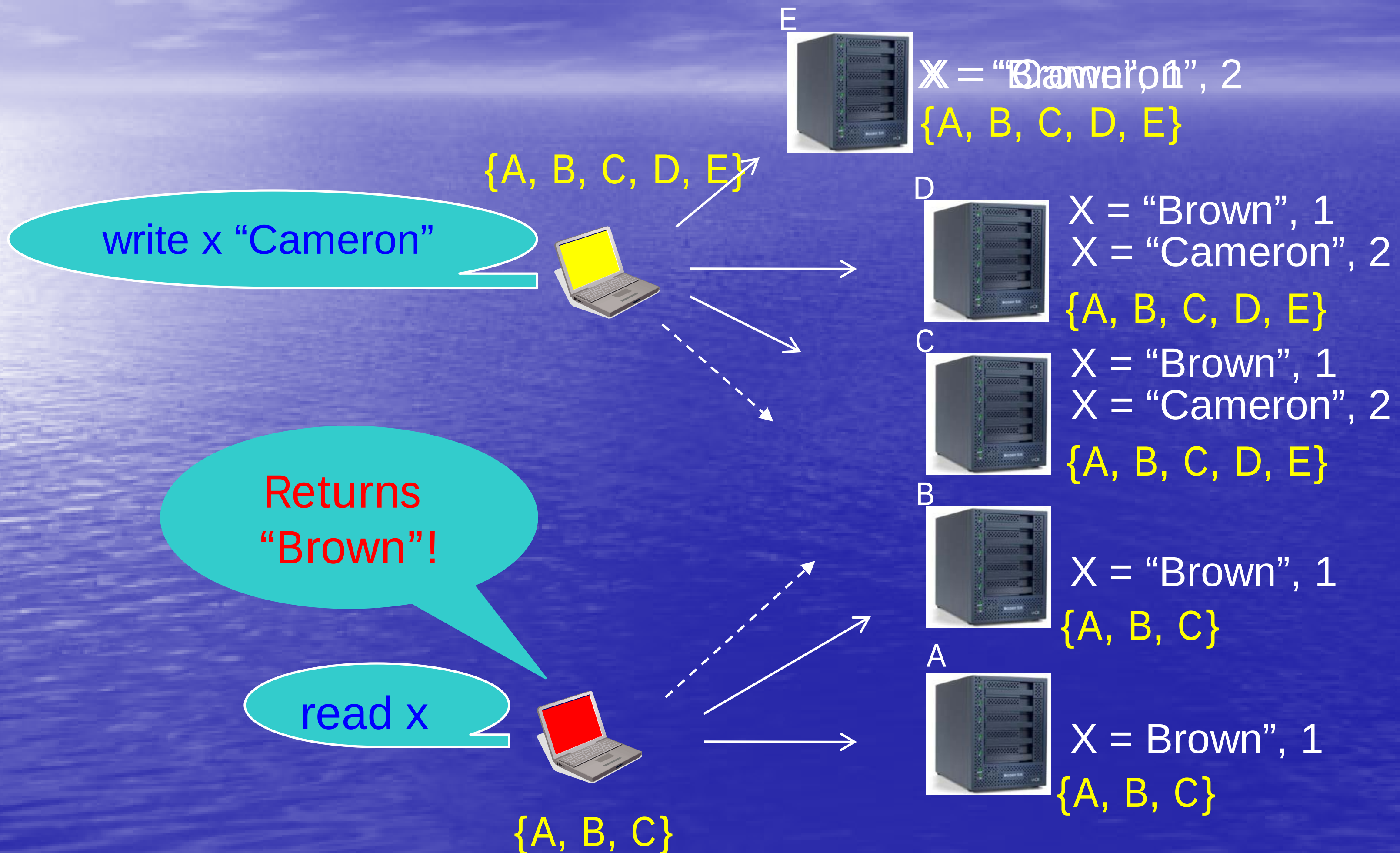


Reconfiguration essential for long-term availability
The challenge: maintain consistency, reliability

Pitfall of Naïve Reconfiguration



Pitfall of Naïve Reconfiguration



Shown in [*Yeger-Lotem, Keidar, Dolev*, PODC’97]

Reconfiguration Option 1: Centralized

Tomorrow Technion servers will be down for maintenance from 5:30am to 6:45am

Virtually Yours,

Moshe Barak

- Can be automatic
  - E.g., Ursa Minor [Abd-El-Malek et al., FAST 05]
- Single point of failure
  - What if manager crashes while changing the system?
- Downtime

Reconfiguration Option 2: Distributed Agreement

- Initiator requests agreement on reconfiguration from other storage nodes
  - Not data-centric

- Use consensus abstraction
  - Each node provides an input, all non-crashed nodes decide on the same output (one of the inputs)
- In theory, might never terminate [FLP85]
- In practice, we have partial synchrony so it usually works

A Theoretical Note

	Static (No Reconfiguration)	Dynamic Reconfiguration
Consensus	Need partial synchrony [FLP85]	Partial synchrony
Atomic read/write object	Asynchronous solution [ABD95]	? Partial synchrony believed necessary

- This project started by trying to prove it
- To this end, we looked for a specification of service availability

Problem with Typical Specs of Dynamic Systems' Availability

Works if: Partial synchrony &
All the majorities our algorithm uses at any given point in time are available

Dynamic Service Progress Specs

- Current config – at first, initial config
- Faulty(t) – nodes that crashed by time t
- Tracking changes due to reconfig

Condition for service availability:

At any time t, fewer than |Current(t)|/2 nodes
from Current(t) $\cup$ AddPending(t)
are in Faulty(t) $\cup$ RemovePending(t)

Dynamic Progress Specs: A Broader Look

- The specification is problem-independent
  - We used it for a read/write storage service
  - It would be interesting to use it for other dynamic (reconfigurable) services
- We show that the progress condition is sufficient
  - Is it also necessary?

Reconfiguration Option 3: DynaStore

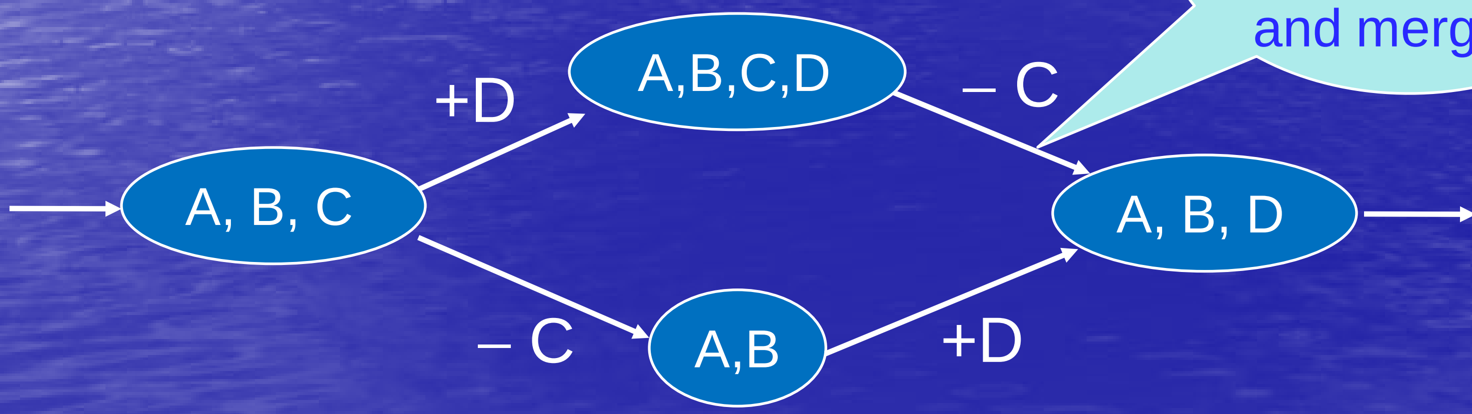
- Satisfy new definition of dynamic service availability
- 1st data-centric distributed reconfiguration
  - With thin storage nodes
- Is partial synchrony (consensus) necessary?

Tracking Evolving Config's

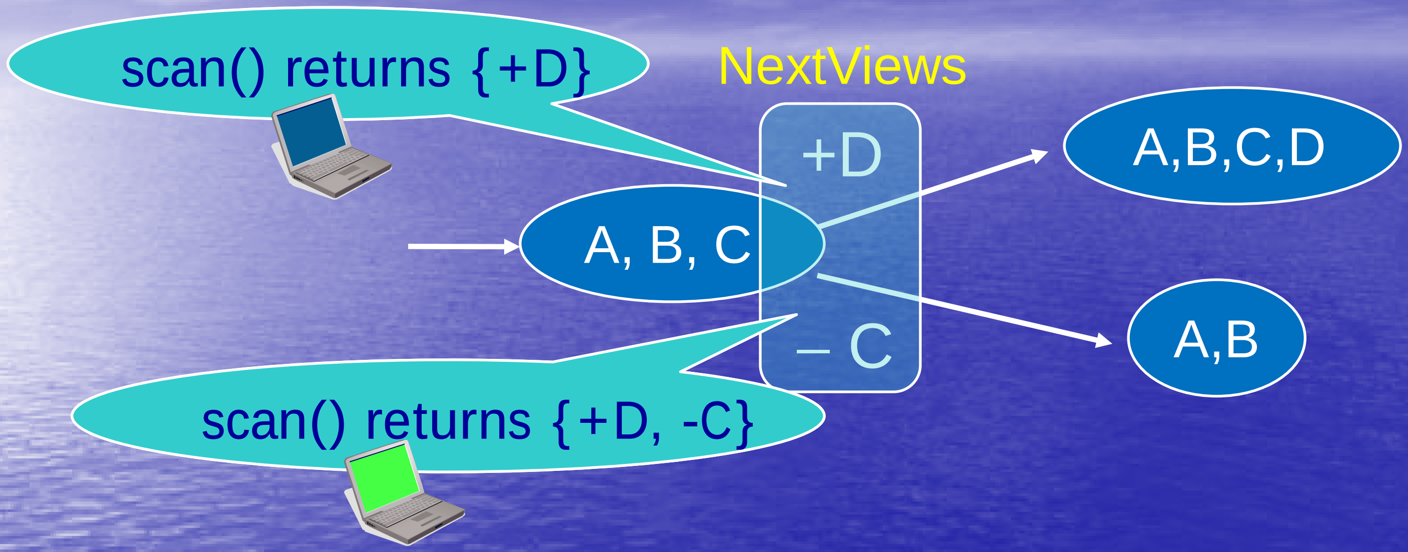
- With consensus: agree on next reconfig

  – Stored at storage nodes

- Without consensus:

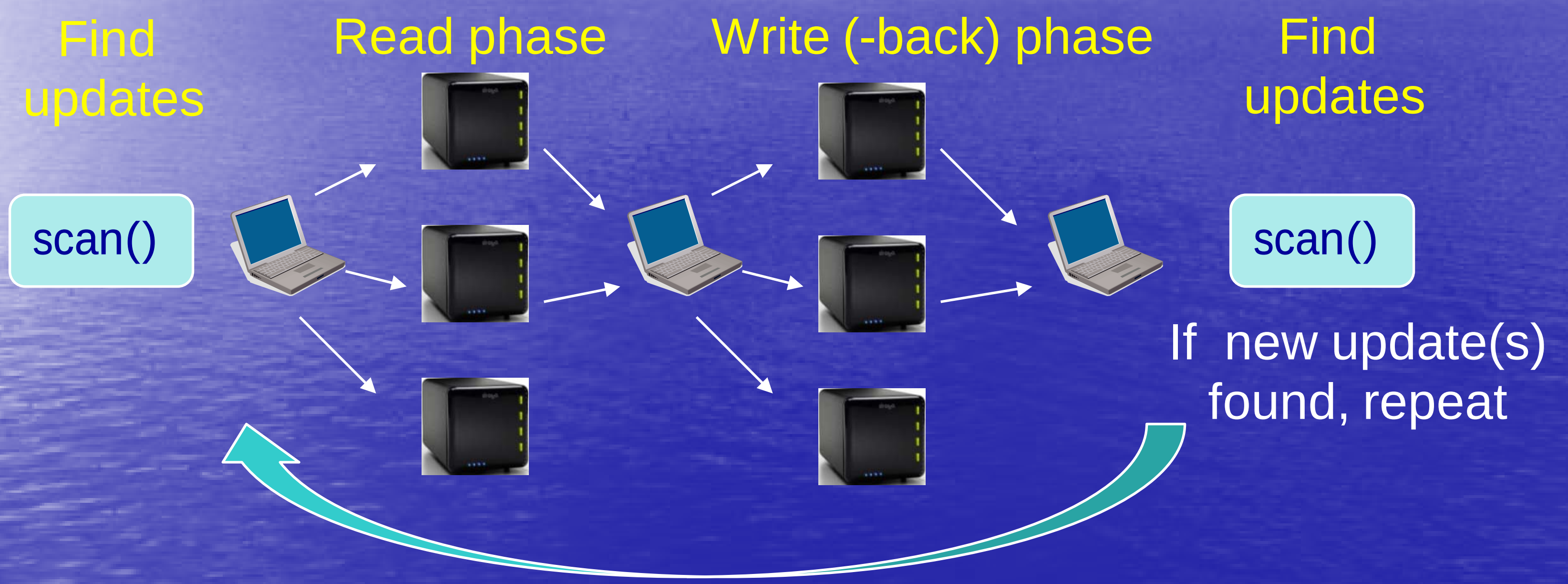


Tracking Config's in DynaStore



- NextViews – Weak Snapshot object
  - Supports update() and scan()
  - All non-empty scans intersect
  - Asynchronous data-centric implementation (see papers)

A Dynamic Reliable Distributed Storage Algorithm (DynaStore)



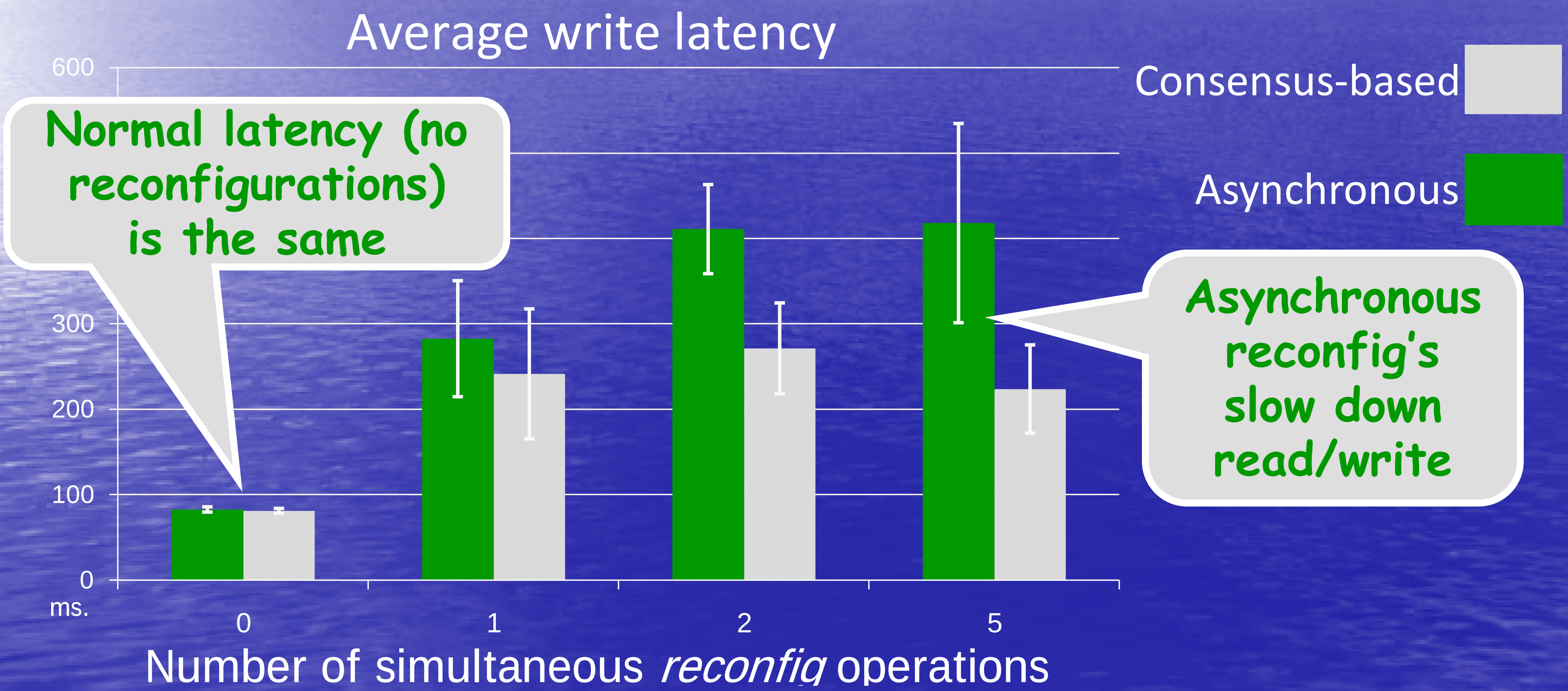
- If scan() finds multiple (concurrent) updates – read/write in all

Consensus-Free Reconfiguration

- It's possible!
  - Dynamic read/write objects "easier" than consensus
  - Works where consensus might not terminate
- But should you do it??
- We experimented to find out….

Just because you can do it doesn't mean you should

The Stronger Progress Guarantees Are Not For Free



Reconfiguration Takeaways

- Reconfiguration is subtle
- Clean service availability definition enables reasoning
- Data-centric distributed reconfiguration is possible with no down time
- *Theoretical angle*:
  Dynamism *per se* does not necessitate agreement
- *Practical implication*:
  Works in more circumstances → more robust
  - But, at a cost

Outline

1. Distributed Storage

   Made up of many cheap, low-reliability storage nodes

   – Achieving reliability, consistency

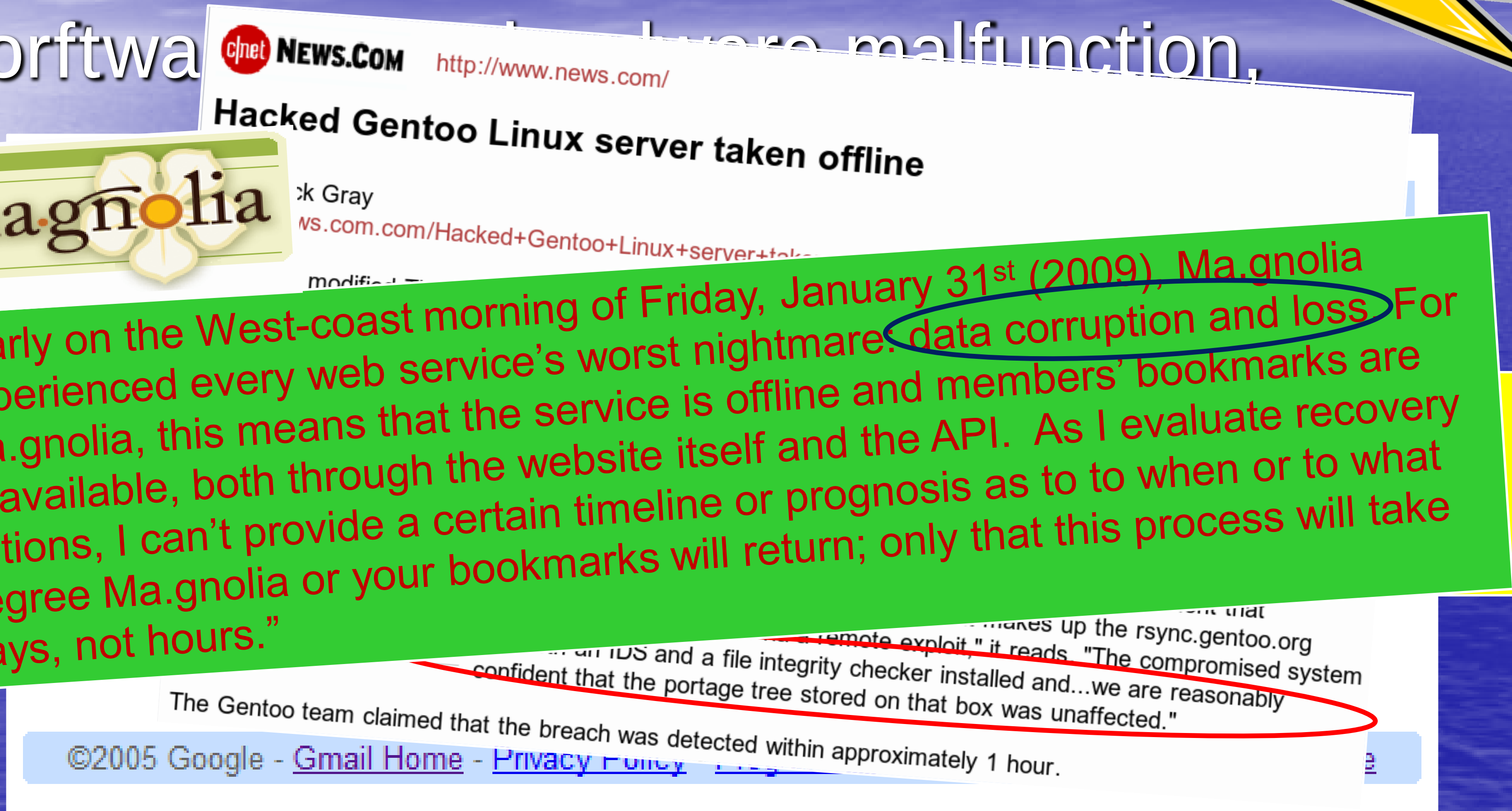
   – The reconfiguration challenge

2. Cloud Storage

   – Can we trust the cloud to ensure reliability and consistency?

Can We Trust The Cloud With Reliability & Consistency?

Sorftwa… malfunction,



"Early on the West-coast morning of Friday, January 31st (2009), Ma.gnolia experienced every web service's worst nightmare: data corruption and loss. For Ma.gnolia, this means that the service is offline and members' bookmarks are unavailable, both through the website itself and the API. As I evaluate recovery options, I can't provide a certain timeline or prognosis as to to when or to what degree Ma.gnolia or your bookmarks will return; only that this process will take days, not hours."

Cachin, Keidar, Shraer: Trusting the Cloud, SIGACT News 2009

Verification for Untrusted Cloud Storage

Cachin, Keidar, Shraer:
Fail-Aware Untrusted Storage, DSN'09

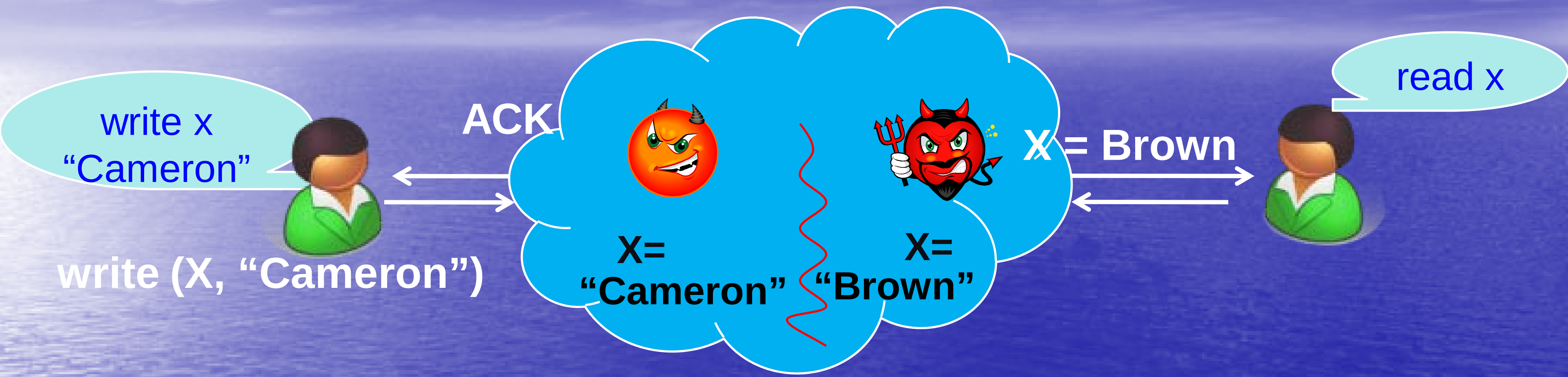
Shraer, Cachin, Cidon, Keidar, Michalevsky, Shaket:
Venus: Verification for Untrusted Cloud Storage

Our Goal

Guarantee reliability and consistency to users of cloud storage

& detect failures

Strong Consistency?



- Impossible to guarantee strong consistency
  – Unless clients communicate directly to complete each operation…
- What can be guaranteed ?

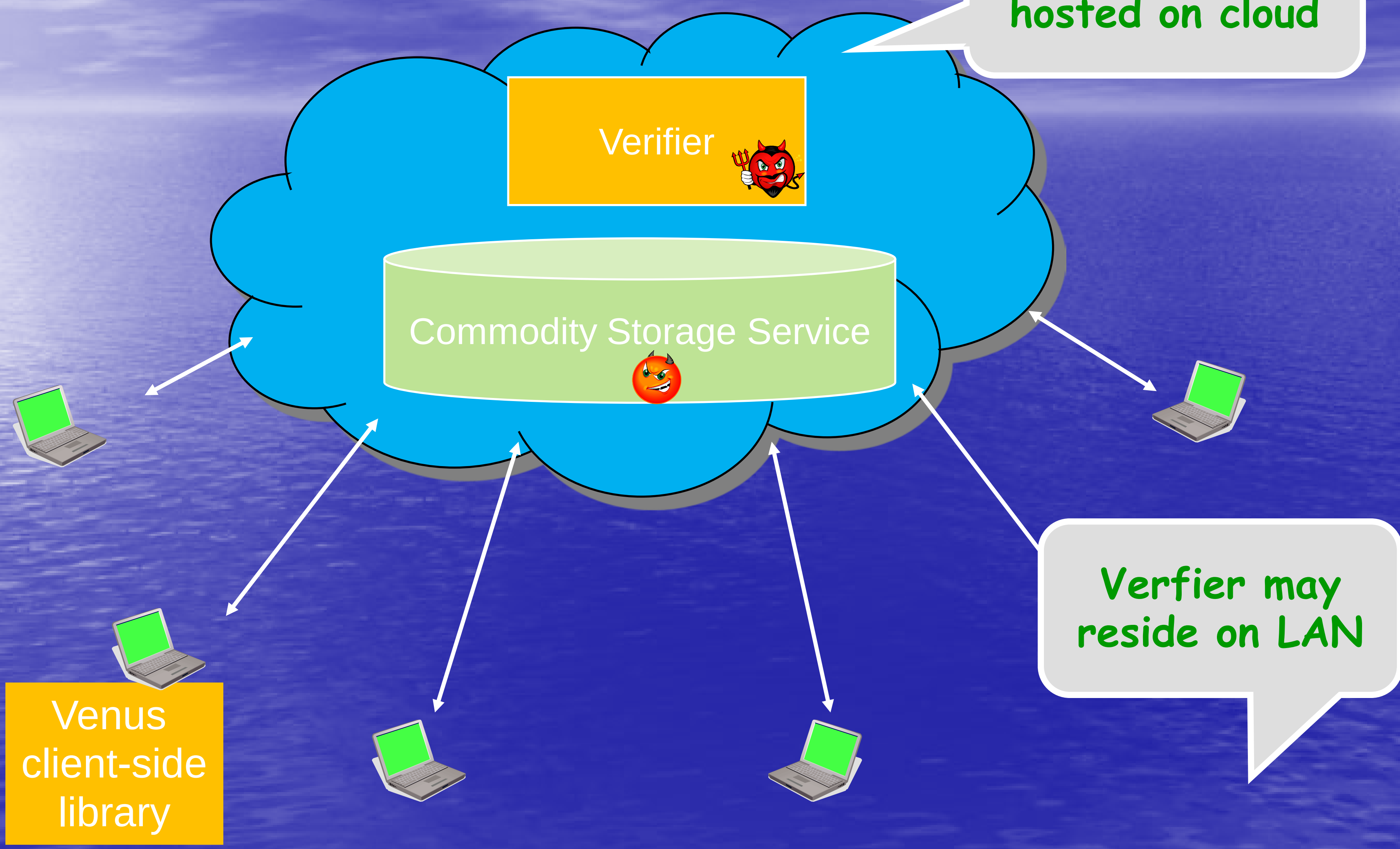
Eventual Consistency Semantics

- Client operations complete optimistically
- Client notified when its operation is known to be consistent
  - But may invoke other operations without waiting for these notifications
- Semantics provided by distributed storage
  - Bayou (SOSP'95)
  - Today in commercial systems, e.g., Amazon's Dynamo (SOSP'07)
- Resembles Futures, Promises, etc.
  - Future<T>: result of an asynchronous computation

Venus Design Principles

1. Defenses should not affect normal case
   – Never block when storage is correct
2. Provide simple, meaningful semantics
   – Eventual consistency
   – Fail awareness – clients learn of every consistency violation
3. Deploy on standard cloud storage
   – Our experiments use Amazon S3

Venus Architecture



Venus Availability

- Operations complete (optimistically) whenever the cloud is online
- Consistency notifications depend on other clients
- Clients may crash, disconnect, but
- Some clients are designated as “core” set

Condition for availability:

Fewer than half the core set clients permanently crash

Venus Basics

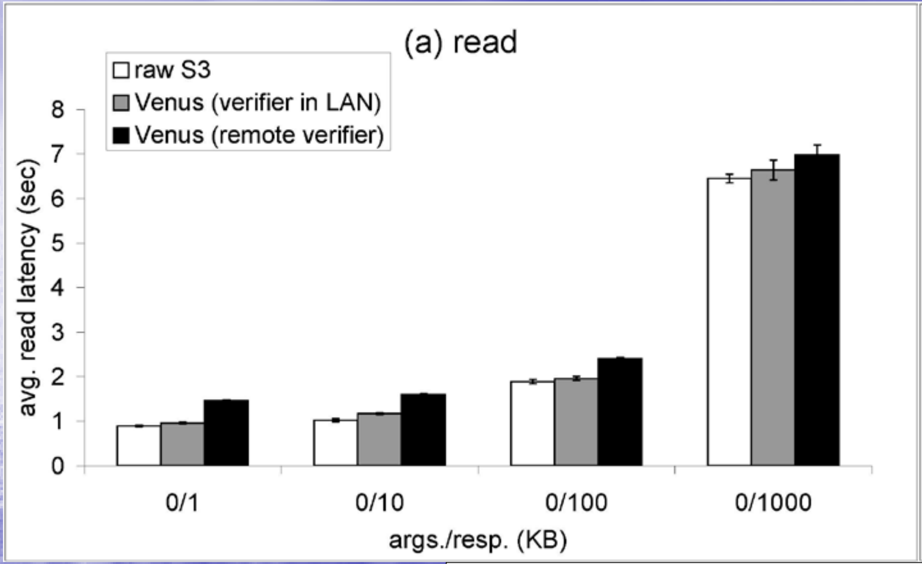
- Read/write data on commodity storage
- Store meta-data (context info) on verifier
  - – Parallelized with data access
- Operations complete optimistically
- Become green when consistent context info is collected from majority of core set
  - – Periodically retrieve context info from verifier
- If no new info for too long, contact other clients
- If context is inconsistent, report error

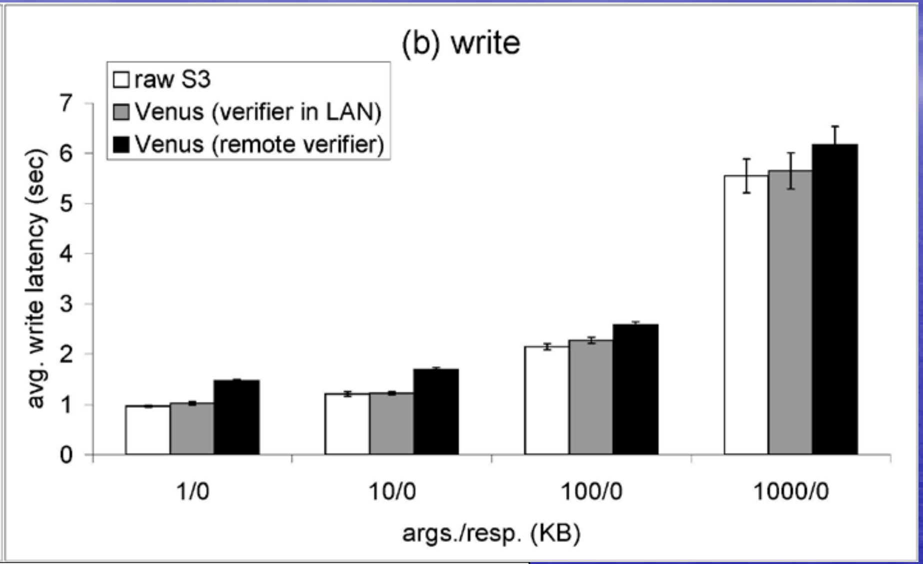
Did core set clients observe my op as I did ?

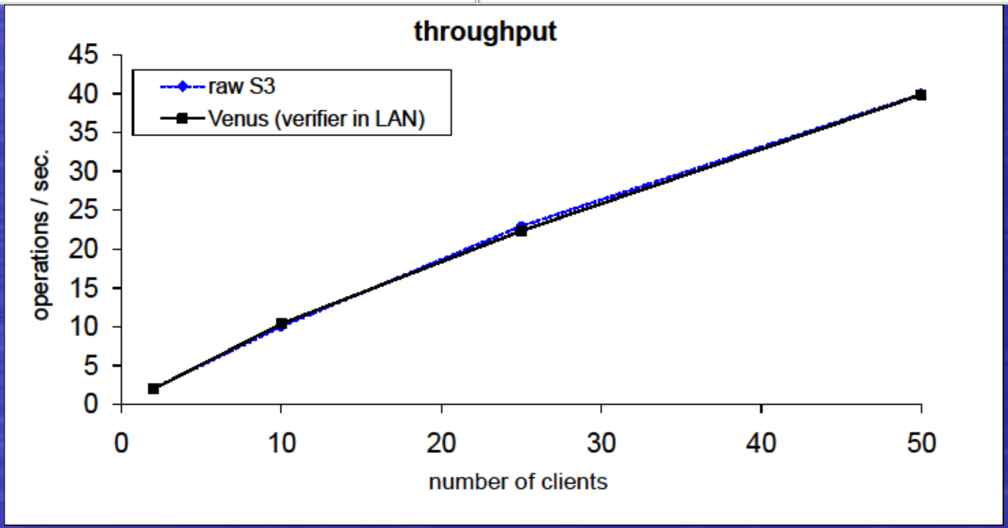
Toolbox

- Clients sign all messages
  - server can’t forge operations, just reorder & hide
- Representation of operation context (V, M):
  - V - vector clock
  - M - vector of aggregate history hashes
    - If op1 is the last operation of client k that op2 observes, k-th entry stores hash of the history op2 is expected to have
- (V, M) pairs compared to determine if two ops are consistent
- Under the hood - “weak fork” consistency
  - Key to being non-blocking
- 12-Page correctness proof

Venus for Amazon S3 vs. Raw S3







Conclusions: Distributed Storage

- There are alternatives to enterprise storage
  - Built from cheap components
    or pay-as-you-go cloud storage
- There are challenges
  - Fault-tolerance
  - Consistency
  - Availability
- But there are also solutions
  - I covered only a few of them today
- Early adopters: companies with big data centers
- Will they become mainstream?

Computing Predictions &Trends

- In 1950s Asimov stories:
  - Multivac supercomputer, 100 sq. miles
- Mocked for decades
  - As ICs became smaller
- And now?

Thanks!

Alex Shraer

Marcos Aguilera, Christian Cachin,
Asaf Cidon, Gregory Chockler,
Rachid Guerraoui, Dahlia Malkhi,
J-P. Martin, Yan Michalevsky,
Dani Shaket, Marko Vukolic

http://webee.technion.ac.il/~idish